การเขียนโปรแกรม ด้วยภาษา C#

Exceptions

Exceptions

Exceptions

ในการพัฒนาโปรแกรม จะมี error เกิดขึ้นได้ 2 แบบ

1. Compilation error
    - การผิดพลาดลักษณะนี้ Compiler จะแจ้งเตือน
2. Run time error
    - การผิดพลาดลักษณะนี้ Compiler จะไม่แจ้งเตือน
    - จะเกิดการ crashes ของโปรแกรมขณะรัน
    - การ crashes ของโปรแกรมเรียกว่าเกิด Exception

Exceptional Circumstances

- Exceptions จะเกิดเมื่อโปรแกรมของเราทำงานผิดพลาด
  - พยายามแปลง invalid string
  - พยายามเข้าถึงสมาชิกอาเรย์ที่อยู่นอกขอบเขต
  - พยายามใช้งานตัวแปร reference ที่เป็น null
  - ...และอีกหลายความพยายาม...
- ภาษา C# มีกลไกรองรับ เพื่อตรวจจับ exceptions ต่างๆ

What is an exception?

- Exception เป็นข้อมูลจำเพาะชุดหนึ่ง ที่บอกว่าโปรแกรมของเรามีความผิดพลาดตรงไหน
- เรามักเจอในโปรแกรมที่ทำงานผิดพลาด
- เราอาจใช้ข้อมูลเหล่านั้นเพื่อหาที่ผิดพลาดในโปรแกรมขณะทำงาน
- Exceptions สามารถเกิดได้จากโปรแกรม หรือ ระบบปฏิบัติการ (การสั่งให้เกิด exception เรียกว่า “thrown”)
- ถ้าไม่มีการเขียนโปรแกรมรองรับ exception (ทางเทคนิคเรียกว่า “caught”) โปรแกรมเราจะ crashes

ตัวอย่าง สาเหตุของ exceptions

```
int i;
i = int.Parse("One");
```

- โปรแกรมนี้ ต้องการ string ที่ประกอบด้วยตัวเลข 0-9 เพื่อจะไปแปลงเป็นจำนวน integer
- แต่ parameter ที่ส่งให้ Parse กลับเป็นตัวอักษร

Catching exceptions

```
try
{
   i = int.Parse("One");
}
catch
{
   Console.WriteLine("Invalid number");
}
```

- ถ้าโปรแกรมในบล็อก try ทำงานล้มเหลว จะเป็นการสร้าง exception
- โปรแกรมจะกระโดดมาทำงานใน catch
  - โดยโปรแกรมของเราไม่ crashes

Exception objects

- ส่วนใดๆ ในโปรแกรมที่น่าสงสัยว่าจะเกิด exception เราสามารถล้อมรอบด้วย try
  - เช่นการเขียนอ่านไฟล์, การเข้าถึง network, การใช้งาน database
- เราสามารถเขียน โปรแกรมไว้ดัก exception ต่างๆ ได้ตามต้องการ
- การดัก exception ใช้บล็อกของ catch

Catching exception details

```
try
{
   i = int.Parse("One");
}
catch (Exception e)
{
   Console.WriteLine(e.Message);
   Console.WriteLine(e.StackTrace);
}
```

- หากต้องการดูรายละเอียดของ exception เราสามารถใช้ exception object มาแสดงรายละเอียด

Finally

- บางครั้ง เราอาจต้องการให้โปรแกรมเราทำงานอย่างปกติไม่ว่าจะมี exception เกิดขึ้นหรือไม่
  - การเชื่อมต่อ network ไม่สำเร็จ ต้องทำการยกเลิกการเชื่อมต่อ
  - การเปิดไฟล์แล้วเขียนไม่สำเร็จ ต้องปิดไฟล์นั้น
  - ฯลฯ
- โครงสร้าง try-catch จะมี finally ไว้ให้ทำในสิ่งที่เราต้องการ
  - คำสั่งใน finally จะถูกเรียกเสมอ ไม่ว่าจะเกิด exception หรือไม่

The Finally Clause

```
try
{
   // Code ที่อาจเกิด exception
}
catch
{
   // Code ที่จัดการ exception
}
finally
{
   // Code ที่ต้องทำงานเสมอ ไม่ว่าจะเกิด exception หรือไม่ก็ตาม
}
```

Exception Types

- Exception
- IndexOutOfRangeException
- SEHException
- SystemException
- NullReferenceException
- AccessViolationException
- ArgumentException
- InvalidOperationException
- ExternalException
- ArgumentNullException
- ArgumentOutOfRangeException
- frlrfSystemRuntimeInteropServicesCOMExceptionClassTopic

สร้าง exceptions เองได้ไหม?

- เราสามารถสร้าง exception ขึ้นมาเองในโปรแกรม

```
throw new Exception ("Bang");
```

- เมื่อโปรแกรมทำงานถึงบรรทัด throw และไม่ได้เขียน catch จะเกิดอะไรขึ้น?

มารยาทในการใช้งาน Exception

- ในการใช้งาน Exception ควรใช้เมื่อจำเป็นจริงๆ
  - เช่นเมื่อตรวจพบว่าโปรแกรมมักจะทำงานผิดพลาดในตำแหน่งนั้น
  - เมื่อต้องใช้ทรัพยากรที่คาดว่าอาจจะไม่มีอยู่ หรือสามารถเคลื่อนย้ายได้
    - เช่น ไฟล์บน thumb drive, การเชื่อมต่อฐานข้อมูล, ฯลฯ

สรุป

- เราใช้ exception เพื่อให้โปรแกรมรันผ่านพื้นที่เสี่ยง โดยที่ไม่เกิดการ crashes ขึ้นกับโปรแกรมของเรา
- exception มีหลายประเภทให้เลือกใช้งาน ดูคู่มือ .NET
- คำสั่งเกี่ยวกับ exception ได้แก่ try-catch-finally